Comput. Math. Applic. Vol. 16, No. 10/11, pp. 967–981, 1988
Printed in Great Britain.
0097-4943/88 $3.00 + 0.00

OPTIMAL ANALYSIS OF LARGE COMPLEX WATER RESOURCES CONVEYANCE SYSTEMS VIA NONSERIAL DYNAMIC PROGRAMMING

A. O. ESOGBUE
School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0205, U.S.A.

C. Y. LEE
Korea Institute of Technology, Taejun, Korea

Abstract—We present a methodology, based on nonserial dynamic programming, for modeling and analyzing complex nonserial converging branch networks. Our algorithm is exemplified via a problem encountered in the design of branched sewer systems in water resources systems. The computational complexity of our approach is shown to be superior to the usually employed discrete differential dynamic programming method.

1. INTRODUCTION

Water resources planning is a multi-faceted multi-staged, continuous process which occurs at several levels and in various locations. At the state, regional and even local levels the systems under consideration are more often than not large scale in nature. That is, planning whether for design, operation or maintenance relates to a system of units rather than a single unit. Thus, cost effectiveness considerations require the treatment of all systems units as a whole. In general, in such large scale systems, the number of variables and alternatives that must be considered forces the planner or analyst using classical approaches favored by practicing engineers to eliminate a large number of possible alternatives. This is done in order to focus on the few that are considered most promising. Only very few experienced engineers can use such a trial and error approach in combination with good judgment to produce cost-effective designs, most of the time, in the usual time and resource constrained design environment. The use of mathematically reliable models, especially those that can be automated has tended to minimize the problems inherent in traditional practices.

As documented in the literature, systems and optimization based approaches have become a useful tool of the modern design engineer. Since its development by Bellman and later by numerous authors [1], dynamic programming has become a very attractive modeling and design tool. However, because of the well-known but perhaps somewhat exaggerated problem of dimensionality, its utility to the practicing engineer has been quite limited. Various authors and model developers have sought approaches to circumvent this problem. Unfortunately, the casualty is usually the problem. Oversimplification and sometimes sensible decomposition methods have been advocated. We have erstwhile postulated that these problems can only be eliminated or more realistically ameliorated when large computers of the super variety, efficient algorithm geared towards memory reduction, parallel computing and above all adroit problem formulation which *ab initio* requires a minimal number of state variables are efficiently utilized to address a given problem. Some of the foregoing perquisites are beginning to be made available to the systems designer.

Our principal contribution is in the area of modeling and computational technology, but specifically nonserial dynamic programming. The purpose is to show how a problem which naturally occurs as a nonserial system but which has hitherto been approximated as a classical serial dynamic program can be directly assaulted via nonserial dynamic programming. This approach naturally minimizes approximation erros and, *ipso facto*, increases accuracy of results. Most of all, it benefits from the global optimality characteristic of classical dynamic programming.

2. THE PROBLEM

A gamut of problems in water resources but particularly those involving water conveyance systems design, multi-basin capacity expansion projects, optimal operation of reservoirs located on different streams in a region, and storm water systems design are large scale in nature. Further, they occur as nonserial, branched systems problems. Most of them, however, have hitherto been treated as serial ones. When nonserial methods have been used only very simple classical structures such as diverging and converging systems have been utilized—and even then the algorithms have been inefficient and computationally unattractive. We have recently developed efficient and even high level computing algorithms for processing various complex nonserial systems [2] and naturally wish to apply them to water resources problems where their use will prove beneficial.

We consider a few examples. Larson and Keckler [3] present a state increment dynamic programming analysis of a four reservoir operation problem (Fig. 1) which is an example of a highly simple converging branch system. They however, do not use nonserial dynamic programming to treat their problem. Instead they solve a four state variable dynamic program with the level of water in each reservoir considered a state variable. Hall and Shephard [4] treat a more realistic reservoir operation problem in the California Central Valley Project consisting of at least four reservoirs with power generation capacities and an assortment of dams, lakes, etc. located on separate streams but converging at a node in the delta. Again, that is a good example of a multi-converging branch system which was analyzed by a combination of linear and dynamic programming by Hall and Shephard [4], Becker and Yeh [5], and Becker *et al.* [6]. Real-time hourly operation problems for the same complex multi-purpose reservoir system were considered by Yeh *et al.* [7, 8]. In each of the foregoing, however, decomposition and approximation methods were used in conjunction with serial dynamic programming—in some cases differential dynamic programming as the optimization procedure. Although useful solutions were obtained, computational complexities and inefficiencies characterize the approach and thus make their use costly.

The approaches proposed in the Central Valley Project by Yeh *et al.* [7] (Fig. 3) have been modified and applied to other important basins in various parts of the United States. For example, the Tennessee Valley Authority implemented Yeh *et al.* [8] incremental dynamic programming and successive approximation model for their real-time optimal scheduling of water releases for flood control and hydro-electric power generation problem. The Texas Water Development Board has considered the use of optimization and computerized models for various water resources problems. A classic problem of concern is the optimal capacity expansion model for surface water resources systems in multi basins. Some examples of these multi-basin networks are given in Figs 4 and 5 for the White River Basin and the San Antonio-Guadalupe River System, respectively. The first

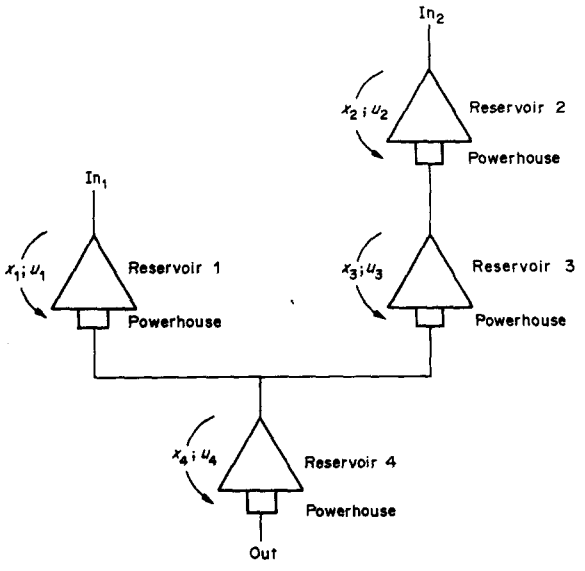

Fig. 1. Network configuration of a four-reservoir problem considered by Larson and Keckler [3].

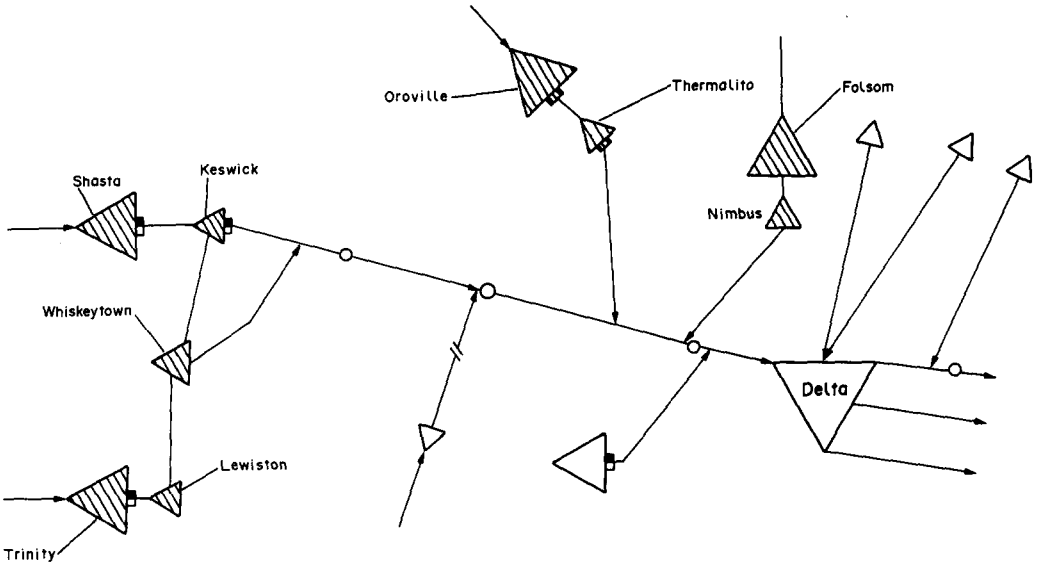

Fig. 2. Operation studies of the California CVP studied by Hall and Shephard [4].

is an example of a simple converging branch system while the second illustrates a more complex multi-converging branch network.

In different problem areas such as sewer systems design, but particularly least cost design of urban storm sewers, the basic problem is a classic example of a complex nonserial network. Modern optimality based approaches, a departure from simulation and heuristic methods, employ dynamic programming but usually discrete differential dynamic programming—but again of the serial variety. Tang *et al.* [10] for example, describe the use of dynamic programming in the design of stormed sewer systems.

A paper of great interest and one that testifies to the need of our approach is due to Mays and Yen [11]. After pointing out the advantages of dynamic programming over other methods, they treat a multi-converging branch sewer system rightfully using a nonserial dynamic programming

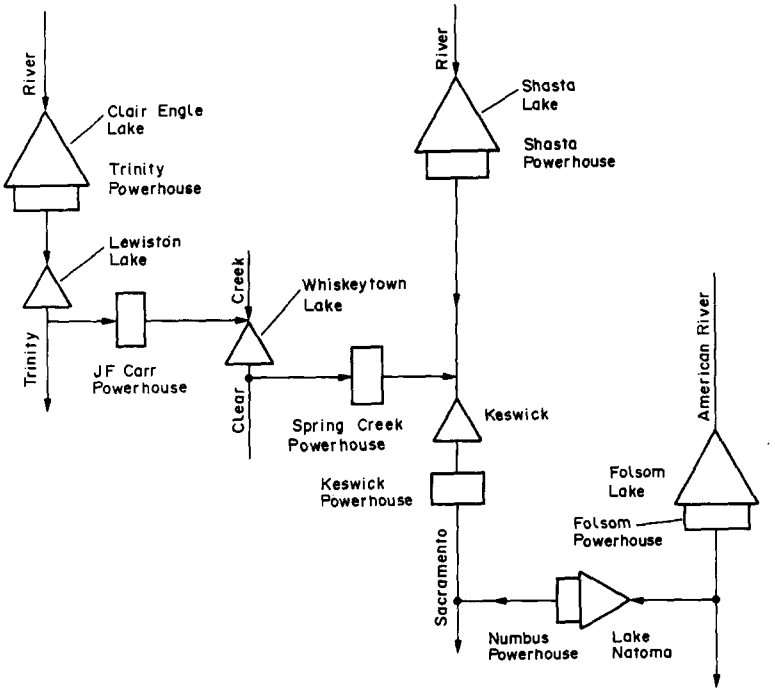

Fig. 3. The California Valley Project System (northern portion) considered by Yeh *et al.* [7].

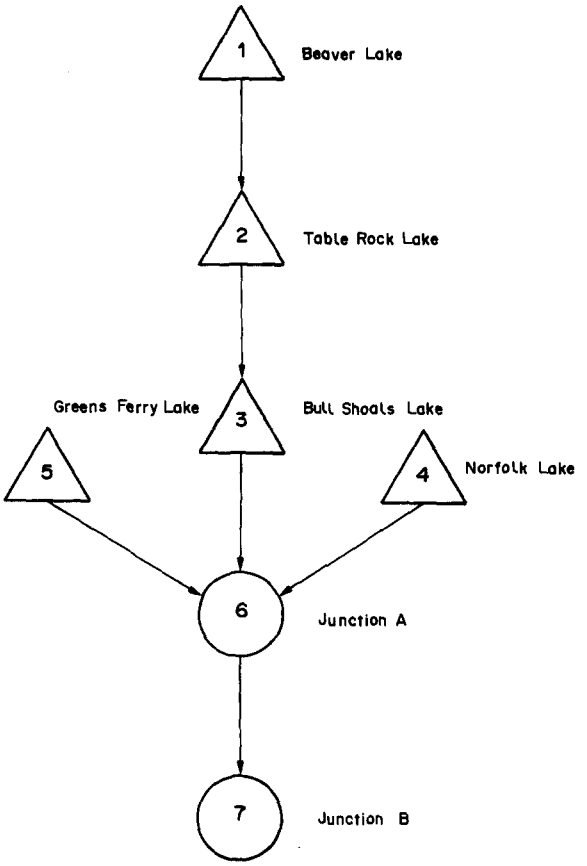

Fig. 4. Network representation of White River system considered by Martin [9]. River Channel (→); reservoir (△); stream junction ○.

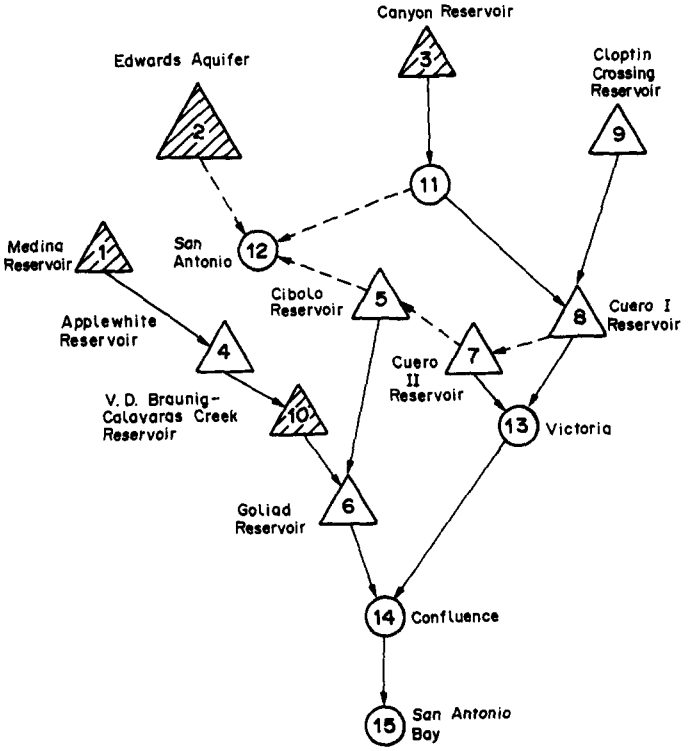

Fig. 5. Network representation of the San Antonio-Quadalupe river system studied by Martin [9]. River channel (⟶); pipeline route (- - →); proposed reservoir (▲); existing reservoir (△).

approach. Their method however has been known to be grossly inefficient [10] and thus not surprisingly was dropped in favor of a discrete differential dynamic programming (DDDP) method. Although DDDP proved to be computationally more attractive—savings in computer time, for the example treated, the accuracy and memory requirements were approximately the same. However, Chow *et al.* [12] show that for large systems DDDP is superior both from the memory and time standpoints. Mays and Yen [11] clearly point out the deficiencies of DDDP including its sometimes local optimality and other computational difficulties, especially when incorporating realistic hydraulic characteristics and configurations. They conclude that "ramification and improvement of the optimization techniques to other branched systems is highly desirable". This dilemma is partly responsible for our work in nonserial dynamic programming and the application reported here.

3. METHODOLOGY FOR OPTIMAL ANALYSIS OF CONVERGING BRANCH NONSERIAL DYNAMIC PROGRAMMING

In a converging branch system as illustrated in Fig. 6, the transition and return at the junction stage, s are triple input functions given by

$$x_{s-1} = t_s(x_{01}, x_s, d_s) \tag{1}$$

$$r_s = r_s(x_{01}, x_s, d_s). \tag{2}$$

At all other stages, the usual transformations of serial dynamic programming are used for the main and branches:

$$x_{n-1} = t_n(x_n, d_n), \qquad n = 1, \ldots, N, n \neq s, \tag{3}$$

$$x_{m-1,1} = t_{m1}(x_{m1}, d_{m1}), \quad m = 1, \ldots, M. \tag{4}$$

In addition, the stage returns are defined as

$$r_n = r_n(x_n, d_n), \qquad n = 1, \ldots, N, n \neq s, \tag{5}$$

$$r_{m1} = r_{m1}(x_{m1}, d_{m1}), \quad m = 1, \ldots, M. \tag{6}$$

Thus the optimization of a single converging branch system can be formulated as follows:

$$\max_{\substack{d_1,\ldots,d_N \\ d_{11},\ldots,d_{M1}}} \sum_{\substack{i=1 \\ \neq s}}^{N} r_n(x_n, d_n) + r_s(x_{01}, x_s, d_s) + \sum_{m=1}^{M} r_{m1}(x_{m1}, d_{m1}), \tag{7}$$

s.t.

$$x_{n-1} = t_n(x_n, d_n), \qquad n = 1, \ldots, N, n \neq S, \tag{8}$$

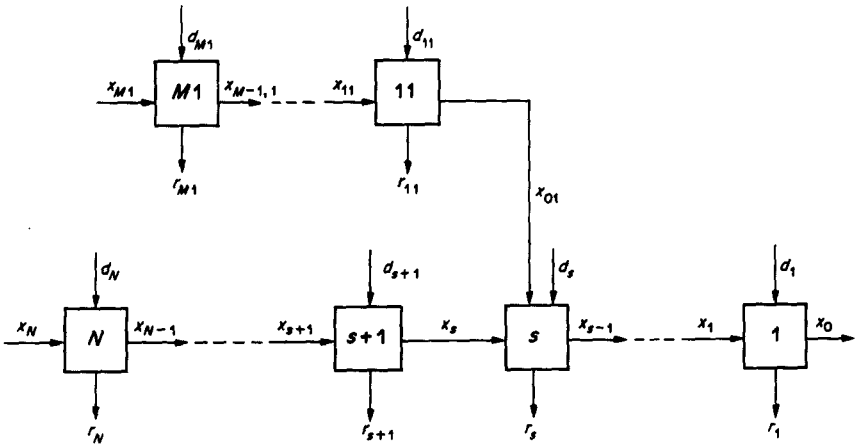

Fig. 6. A single converging branch system.

$$x_{s-1} = t_s(x_{01}, x_s, d_s), \tag{9}$$

$$x_{m-1,1} = t_{m1}(x_{m1}, d_{m1}), \quad m = 1, \ldots, M, \tag{10}$$

$$x_n \in X_n, \qquad d_n \in D_n, \qquad x_{m1} \in X_{m1}, \qquad d_{m1} \in D_{m1}, \qquad \forall n, m. \tag{11}$$

Note that in the foregoing formulation, at the junction stage s the stage return r_s is a function of input variables x_{01}, x_s and the stage decision d_s. Since the branch output x_{01} is a function of the branch input x_{M1} and the decisions $(d_{11}, \ldots, d_{M1})$, the decision at the the converging branch affects the return from the main serial process. Thus, the converging branch can not be optimized independently of the main serial process as in the case of the diverging branch nonserial network. In general, the converging branch is treated as an initial–final value problem (often termed a two-point boundary value problem); this therefore results in a two-dimensional optimization problem.

The main serial system is optimized as the usual serial dynamic programming process up to stage $s-1$. At the junction stage s, however, the s-stage return combines the branch return $f_{M1}(x_{01}, x_{M1})$ with the $(s-1)$-stage optimal return $f_{s-1}(x_{s-1})$ and the return at stage s. Thus, the s-stage return function is given by

$$f_s(x_s, x_{M1}) = \max_{x_{01}, d_s} [r_s(x_{01}, x_s, d_s) + f_{s-1}(t_s(x_{01}, x_s, d_s)) + f_{M1}(x_{01}, x_{M1})]. \tag{12}$$

Aris, *et al.* [13] suggest that x_{01} be treated as a "cut state". They choose a particular value for the branch output x_{01}, and use the boundary value optimization to obtain the optimal branch output $f_{M1}(x_{01}, x_{M1})$. Simultaneous selection of d_s would, for a given value of x_s, determine a total return in equation (12). Once this quantity has been stored, together with the corresponding decision, new values of x_{01} and d_s can be chosen by a direct search method. Repetition of this guided search eventually gives $x_{01}^*(x_s)$ and $d_s^*(x_s)$ the optimal values of x_{01} and d_s for every value of x_s. This is not difficult to do if the branch input x_{M1} is a constant. Notice, however, that if x_{M1} is not a constant, then the branch input x_{M1} also has to be treated as another "cut state" which results in the following three-decision optimization problem:

$$f_x(x_s) = \max_{\substack{x_{01}, x_{M1} \\ d_s}} (r_s(x_{01}, x_s, d_s) + f_{s-1}(t_s(x_{01}, x_s, d_s)) + f_{M1}(x_{01}, x_{M1})]. \tag{13}$$

This is clearly a difficult problem from the standpoint of computational complexity. We have nevertheless developed an algorithm in which this three-decision optimization problem has been reduced to three one-decision problems which are much easier to solve. The details of a high level computing version of this algorithm are presented in Esogbue and Warsi [14].

We now illustrate the above with an example, and then focus our attention to the development of a methodology for the analysis and design of complex multi-converging branch systems which may prove helpful in analyzing real life water resource systems.

Example 1

To illustrate consider a single converging branch system as shown in Fig. 7 with the following transition and return functions and restrictions on the variables:

$$t_n = x_n + d_n, \qquad n = 1, 2, 4, 5,$$

$$t_3 = x_3 + x_{01} + d_3$$

$$t_{m1} = x_{m1} + d_{m1}, \qquad m = 1, 2,$$

$$r_n = x_n + d_n^2, \qquad n = 1, 2, 4, 5,$$

$$r_3 = x_3 + x_{01} + d_3,$$

$$r_{m1} = x_{m1} + d_{m1}^2, \qquad m = 1, 2,$$

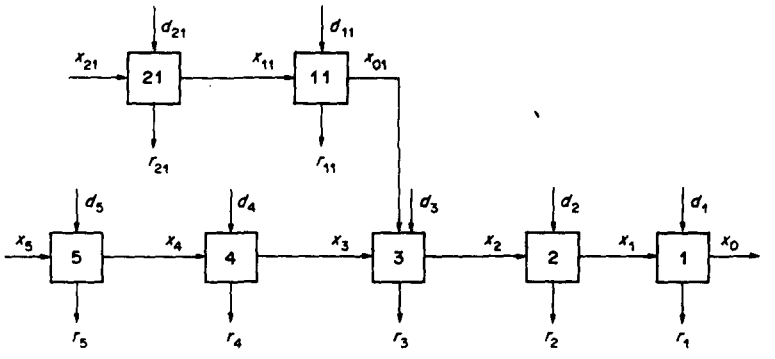

Fig. 7. A converging branch system for Example 1.

$$29 \leqslant x_1 \leqslant 38, \quad 27 \leqslant x_2 \leqslant 36, \quad 5 \leqslant x_3, x_4 \leqslant 14, \quad 0 \leqslant x_5 \leqslant 9,$$

$$16 \leqslant x_{01} \leqslant 20, 10 \leqslant x_{11}, x_{21} \leqslant 19, \quad 0 \leqslant d_n, d_{m1} \leqslant 2,$$

$$x_n, x_{m1}, d_n, d_{m1}: \text{integer}.$$

Note that this simple system contains only one converging branch which consists of two stages, i.e. $M = 2$ while the main chain contains five stages i.e. $N = 5$. The converging node $s = 3$. The algorithm receives input data either in terms of functions or tables.

Table 1 illustrates the computational results with the optimal input, decision and return at each stage of the system, as well as the input tables utilized in the example.

Our algorithm is exceedingly efficient and overcomes the problems associated with previously reported ones as for example Mays and Yen [11]. Its efficiency is exemplified when we consider a multi-converging branch system.

4. AN ALGORITHM FOR MULTI-CONVERGING BRANCH SYSTEMS

We define a multi-converging branch system as a system with more than one converging branch as shown in Fig. 8. Each branch in the figure has M_i stages and converges to a stage s_i in the main system.

When a system has multi-converging branches, it is important to carry out optimizations over branch input $x_{M_i i}$ at each corresponding stage s_i. Otherwise, $x_{M_i i}$ would have to be carried as state variables to be examined exhaustively during the optimization at some stages in the main serial system. More specifically, at each junction s_i of the ith branch the branch input $x_{M_i i}$ has to be kept as a state variable. Thus, at the last junction s_D, the stage return function may be expressed as

$$f_{s_D}(x_{s_D}, x_{M_1 1}, \ldots, x_{M_D D}) = \max_{x_{OD}, d_{s_D}} [r_{s_D}(x_{s_D}, d_{s_D}) + f_{M_D D}(x_{M_D D}(x_{M_D} d, x_{OD})) + f_{s_D - 1}(t_{s_D}(x_{s_D}, d_{s_D}), x_{M_1 1}, \ldots, x_{M_D - 1} d - 1)]. \quad (14)$$

However, when we apply the procedure developed for converging branch systems, the $D + 1$-dimensional optimization problem can be reduced to a sequence of one-dimensional problems for the main serial chain. This is a tremendous reduction of dimensionality which is very useful in applications.

In analyzing the computational demands of the multi-converging branch system, we notice that they are highly affected by the optimization order of the converging branches. If we optimize the branches first, each branch calls for the memory space to store the optimal branch return. However, by employing the method that optimizes and combines branch i when the procedure has reached the corresponding converging state s_i in the main serial process, we can save the memory space for the optimal branch returns. The reduction effect in this multi-converging branch system is greater than in the multi-diverging branch system, since the optimization of a single converging branch is two-dimensional.

Table 1. Computer output for the converging branch system of Example 1

```
DECISION TABLE FOR BRANCH 1
 ****************************
****************************
****************************
****************************
    2. ************************
    1.    2. *****************
    0.    1.    2. ***********
*****     0.    1.    2. *****
***********     0.    1.    2.
******************    0.    1.
    2.    2.    2.    2.    2.
    2.    2.    2.    2.    2.
    2.    2.    2.    2.    2.
    2.    2.    2.    2.    2.
    2.    2.    2.    2.    2.
    1.    2.    2.    2.    2.
    0.    1.    2.    2.    2.
    2.    0.    1.    2.    2.
    2.    2.    0.    1.    0.
    2.    2.    2.    0.    0.

OPTIMAL BRANCH INPUT & RETURN OF BRANCH1
  X01        INPUT    RETURN
   16         14.       34.
   17         15.       36.
   18         16.       38.
   19         17.       40.
   20         16.       42.

"RETURN TABLE FOR MAIN PROCESS"
 33.  34.  35.  36.  37.  38.  39.  40.  41.  42.
 64.  66.  68.  70.  72.  74.  76.  78.  80.  82.
133. 136. 139. 142. 145. 148. 151. 154. 157. 160.
148. 152. 156. 160. 164. 168. 172. 176.   0.   0.
  4.   5.   6. 155. 160. 165. 170. 175. 180. 185.

"DECISION TABLE FOR MAIN PROCESS"
  2.   2.   2.   2.   2.   2.   2.   2.   2.   2.
  2.   2.   2.   2.   2.   2.   2.   2.   2.   2.
  2.   2.   2.   2.   2.   2.   2.   2.   2.   2.
  2.   2.   2.   2.   2.   2.   2.   2.   0.   0.
  2.   2.   2.   2.   2.   2.   2.   2.   2.   2.

OPTIMAL INPUT X01  FROM BRANCH 1  TO JUNCTION  #
 20.  20.  20.  20.  20.  20.  20.  20.  20.  20.

 "OPTIMAL SOLUTION OF THE SYSTEM"

 "STAGE  INPUT  DECIS  RETURN"

 "MAIN SERIAL PROCESS"
    5      9        2       13.
    4     11        2       15.
    3     13        2       35.
    3     20
    2     35        2       39.
    1     37        2       41.

 BRANCH  1  FROM JUNCTION  3
    2     16        2       20.
    1     18        2       22.

 TOTAL OPTIMAL RETURN IS    185.
 THE CPU TIME IN SECONDS IS .235
```

Now, to generalize our analysis, it is instructive to consider a system with branches of the following two classes:

(1) Class I. Branches that converge at the main serial system.
(2) Class II. Branches that do not converge at the main serial system.

Figure 8 and 9 illustrate the branches of these two classes. Notice that in Fig. 8, each branch converges separately and directly at the main chain. In Fig. 9, however, one branch converges at $C1$—a node in another branch. Also, two branches converge simultaneously at node S_2 of the main chain. When a node in the main chain has branches of the two classes, the branch of Class II needs to be optimized prior to that of Class I in order to reduce the computational storage demand.

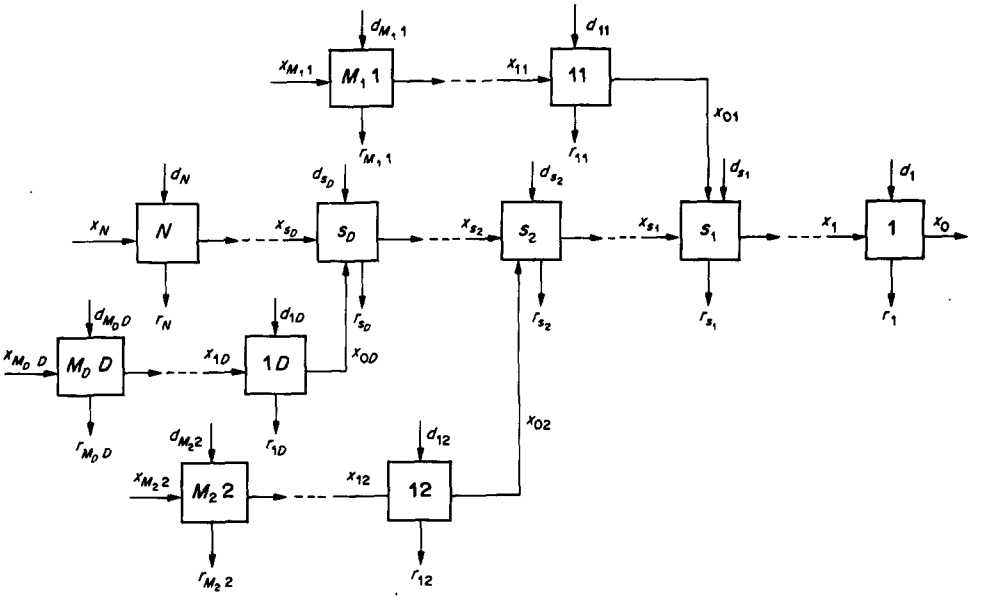

Fig. 8. A multi-converging branch system.

We wish to present a detailed optimization procedure for processing a complex multi-converging branch system of the type illustrated in Fig. 9. The flow chart of the algorithm is also presented in Fig. 10. To aid in understanding our presentation we first define U_n and V_n, respectively, as

(i) the number of branches of Class I that converge into stage n of the main serial process

and

(ii) the number of branches of Class I that are connected to stage n of the main serial process.

4.1 Optimization of the multi-converging branch system

Initialization. Let $n = 0$ and $f_n(x_n) = 0$ and go to Step 1.

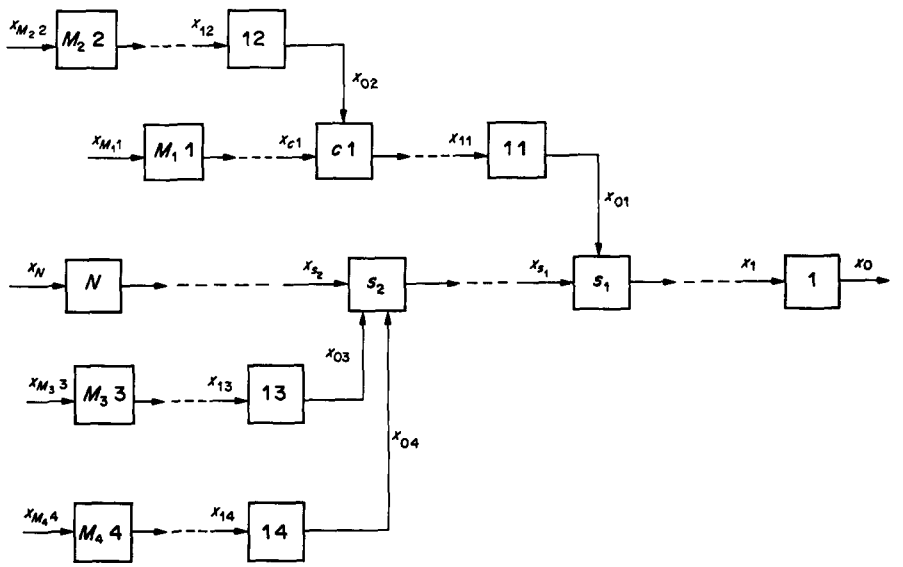

Fig. 9. A complex multi-converging branch system.

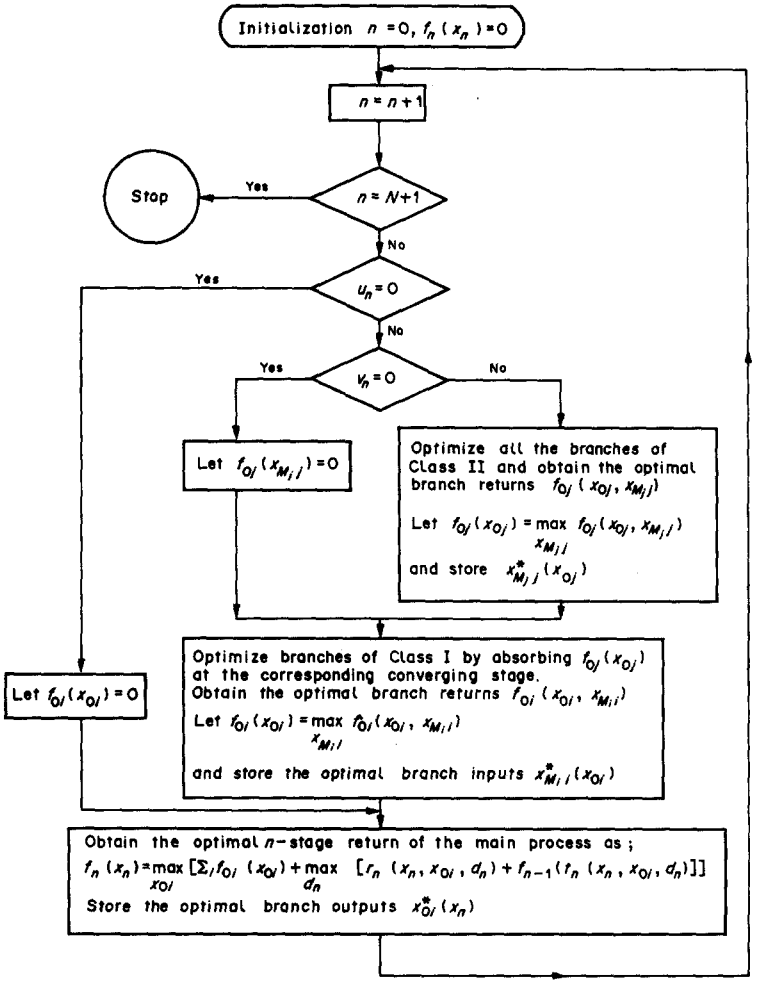

Fig. 10. Flow chart of the multi-converging branch algorithm.

Step 1. Replace n with $n+1$. If $n=N+1$, then stop. The optimal system return $f_N(x_N)$ is obtained. Otherwise, if stage n has any converging branch i of Class I, i.e. $u_n \neq 0$, then go to Step 2.

If $u_n = 0$, let $f_{0i}(x_{0i}) = 0$ and go to Step 4.

Step 2. If stage n has any branch j of Class II, i.e. $v_n \neq 0$, then optimize the branches and obtain the optimal branch returns. $f_{0j}(x_{0j}, x_{M_j i})$.

$$\text{Let } f_{0j}(x_{0j}) = \max_{x_{M_j j}} f_{0j}(x_{0j}, x_{M_j j}).$$

Store the optimal branch inputs $x_{M_j j}(x_{0j})$ and go to Step 3. Otherwise, if $v_n = 0$, then let $f_{0j}(x_{0j}) = 0$ and go to Step 3.

Step 3. Optimize branches of Class I by absorbing $f_{0j}(x_{0j})$ at the corresponding converging stages. Store the optimal branch output x_{0j}. Obtain the optimal branch returns $f_{0i}(x_{0i}, x_{M_i i})$. Let

$$f_{0i}(x_{0i}) = \max_{x_{M_i i}} f_{0i}(x_{0i}, x_{M_i i}).$$

Store the optimal branch inputs $x^*_{M_i i}(x_{0i})$ and go to Step 4.

Step 4. Obtain the optimal n-stage return of the main serial chain by absorbing all the branches as

$$f_n(x_n) = \max_{x_{0i}} \left[\sum_i f_{0i}(x_{0i}(+ \max_{d_n} [r_n(x_{0i}, x_n, d_n) + f_{n-1}(t_n(x_{0i}, x_n, d_n))])) \right].$$

Store the optimal branch outputs $x_{0i}^{*}(x_n)$ and go to Step 1.

This concludes the optimization phase of the multi-converging branch system. In the next section we show how this algorithm can be utilized to solve a complex wter resources problem.

5. OPTIMAL ANALYSIS OF BRANCHED SEWER SYSTEMS VIA THE MULTI-CONVERGING BRANCH ALGORITHM

The problems of designing the size and slope of sewer pipes in a drainage system can be solved by employing the optimization procedure for the multi-converging branch systems. Mays and Yen [11] consider a sewer system illustrated in Fig. 11 where many branches converge into a main stream. The main stream consists of nine sewer pipes and ten manholes. Connected to the main system are four branches that converge into the main stream at manholes 3, 4, 5 and 6, respectively.

The problem is to determine the optimum pipe size and the elevation of upstream and downstream of each pipe such that the installation cost for the sewer system is minimized under several constraints. Mays and Yen solved this problem by applying discrete differential dynamic programming. They change the state space in each iteration by improving the trial trajectory (the sequence of states for different stages). The conventional dynamic programming approach is used within the neighborhood states of the previous trial trajectory. The procedure terminates when the increment of the state is less than a predetermined value.

Following Mays and Yen's problem formulation, we now solve the branched sewer system by applying the optimization algorithm developed in Section 4. First, the following assumptions and constraints are used:

1. Any size of the diameter of the sewer pipe is available that is computed by using the Manning's formula (see equation (11) of Ref. [11]).
2. The diameter of a pipe can not be less than that of the pipe preceding it.
3. The upstream elevation of a pipe is equal to the lowest downstream of the pipes preceding it.
4. A minimum soil cover depth of 8 ft above the crown of each pipe is assumed.

To solve the problem we also define the following dynamic programming variables.

Stages. The stages are analogous to the ordering of manholes in the main stream and the branches, i.e.

$$n = 1, \ldots, 9,$$

$$m = 1, \ldots, M_i, \quad i = 1, \ldots, 4,$$

where $M_1 = M_2 = M_3 = 3$ and $M_4 = 2$.

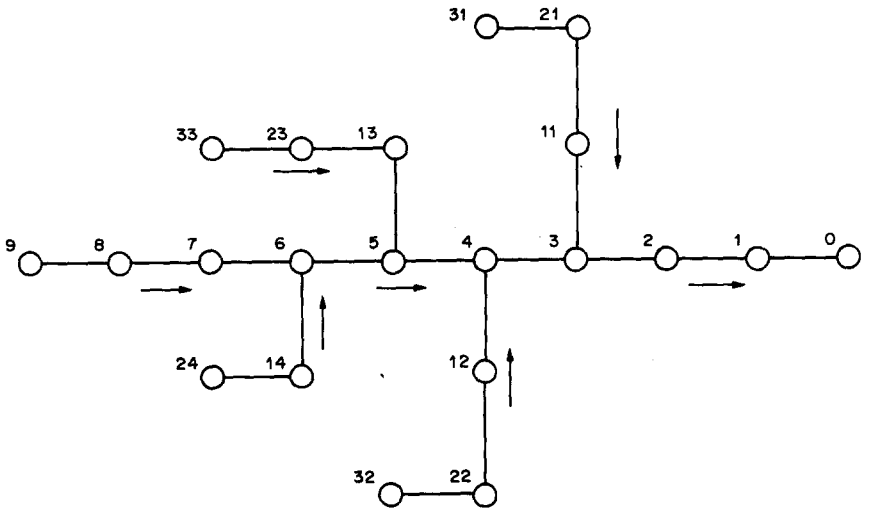

Fig. 11. An example of a sewer system. Manhole (○); sewer pipe (——).

States. The state x_n (or x_{mi}) at manhole n (or mi) is analogous to the crown elevation of the pipes connected to the manhole.

Decisions. The decision d_n (or d_{mi}) is the drop of the pipes across the stage.

Returns. The return r_n (or r_{mi}) is the cost of the installation of the manhole n (or mi) and the pipes connecting it to the manhole $n-1$ (or $m-1, i$).

Transitions. The transition function is defined as

$$x_{n-1} = x_n - d_n, \qquad n = 1, \ldots, 9,$$

$$x_{m-1,i} = x_{mi} - d_{mi}, \quad m = 1, \ldots, M_i, \quad i = 1, \ldots, 4.$$

Now by applying the multi-converging branch algorithm developed in Section 4 (see Fig. 10 for the functional equation at each stage) we optimize the system from manhole 1–9 in the main stream. The optimum return of each branch is computed and combined to the main system at the corresponding junction manhole. Table 2 illustrates the computational results for the problem. At each stage, 11 discretizations were used for the input elevations of each pipe. The physical data and cost functions given in Ref. [11] were used to determine the optimal diameter and the slope of each sewer pipe. The optimal solution is obtained with the outlet elevation of the system to be at 434 ft. Due to the several different assumptions and constraints used, however, the upstream and downstream elevations of the pipes are slightly different from the solution given by Mays and Yen.

Table 2. Computer output of the problem for the sewer system

OPTIMAL BRANCH INPUT & RETURN OF BRANCH 1

XO1	INPUT	RETURN
447.00	460.00	15837.
446.80	460.00	15825.
446.60	460.00	15815.
446.40	460.00	15806.
446.20	460.00	15799.
446.00	460.00	15794.
445.80	460.00	15790.
445.60	460.00	15786.
445.40	460.00	15804.
445.20	460.00	15836.
445.00	460.00	15868.

OPTIMAL BRANCH INPUT & RETURN OF BRANCH 2

XO2	INPUT	RETURN
457.00	477.00	22911.
456.80	477.00	22885.
456.60	477.00	22861.
456.40	477.00	22861.
456.20	477.00	22881.
456.00	477.00	22901.
455.80	477.00	22924.
455.60	477.00	22948.
455.40	477.00	22973.
455.20	477.00	23000.
455.00	477.00	23028.

OPTIMAL BRANCH INPUT & RETURN OF BRANCH 3

XO3	INPUT	RETURN
462.00	482.00	23441.
461.80	482.00	23419.
461.60	482.00	23400.
461.40	482.00	23382.
461.20	482.00	23367.
461.00	482.00	23389.
460.80	482.00	23413.
460.60	482.00	23439.
460.40	482.00	23466.
460.20	482.00	23495.
460.00	482.00	23524.

OPTIMAL BRANCH INPUT & RETURN OF BRANCH 4

XO4	INPUT	RETURN
472.00	482.00	12367.
471.80	482.00	12353.
471.60	482.00	12341.
471.40	482.00	12331.
471.20	482.00	12322.

continued opposite

Table 2—*continued*

```
                    471.00    482.00        12316.
                    470.80    482.00        12310.
                    470.60    482.00        12307.
                    470.40    482.00        12326.
                    470.20    482.00        12365.
                    470.00    482.00        12405.

                  "STAGE RETURN AT EACH STAGE OF MAIN PROCESS"

AT STAGE 1
  37528.74  38394.02  39288.22  40214.15  41175.04  42174.63  43217.29  44308.11
  45453.16  46659.64  47936.30

AT STAGE 2
  68695.39  69779.50  70937.79  72183.35  73532.93  74891.78  76279.62  77699.27
  79153.97  80629.76  82123.32

AT STAGE 3
 104441.86 105076.27 105741.41 106440.77 107178.56 107959.90 108791.09 109679.95
 110638.28 111675.75 112809.01

AT STAGE 4
 148489.61 148710.24 148936.27 149188.91 149444.80 149704.09 149966.94 150233.52
 150504.03 150778.64 151057.59

AT STAGE 5
 188045.28 188285.84 188536.82 188798.85 189072.67 189372.88 189685.26 190011.25
 190352.55 190711.20 191089.64

AT STAGE 6
 212424.59 212500.84 212580.43 212726.15 212887.26 213051.97 213220.26 213392.09
 213569.66 213749.50 213931.72

AT STAGE 7
 218012.27 218071.40 218131.60 218192.94 218255.49 218319.34 218384.58 218451.31
 218557.32 218666.88 218778.85

AT STAGE 8
 223826.38 223886.95 223948.38 224010.72 224074.01 224138.33 224203.72 224270.25
 224337.99 224440.48 224552.56

AT STAGE 9
 228315.70 228377.86 228441.67 228507.29 228574.89 228644.67 228716.87 228791.77
 228869.71 228951.07 229060.20

 "DECISION TABLE FOR MAIN PROCESS"


AT STAGE 1
      5.00      4.80      4.60      4.40      4.20      4.00      3.80      3.60
      3.40      3.20      3.00

AT STAGE 2
      3.00      2.80      2.60      2.40      2.20      2.20      2.20      2.20
      2.20      2.00      2.00

AT STAGE 3
      4.00      3.80      3.60      3.40      3.20      3.00      2.80      2.60
      2.40      2.20      2.00

AT STAGE 4
     10.00      9.80      9.60      9.40      9.20      9.00      8.80      8.60
      8.40      8.20      8.00

AT STAGE 5
      5.00      4.80      4.60      4.40      4.20      4.00      3.80      3.60
      3.40      3.20      3.00

AT STAGE 6
     10.00      9.80      9.60      9.40      9.20      9.00      8.80      8.60
      8.40      8.20      8.00

AT STAGE 7

      7.00      6.80      6.60      6.40      6.20      6.00      5.80      5.60
      5.40      5.20      5.00

AT STAGE 8
      8.00      7.80      7.60      7.40      7.20      7.00      6.80      6.60
      6.40      6.20      6.00

AT STAGE 9
      5.00      4.80      4.60      4.40      4.20      4.00      3.80      3.60
      3.40      3.20      3.00

OPTIMAL INPUT XO1  FROM BRANCH 1  TO JUNCTION  3

    447.00    446.80    446.60    446.40    446.20    446.00    445.80    445.60
    445.60    445.60    445.60

OPTIMAL INPUT XO2  FROM BRANCH 2  TO JUNCTION  4

    457.00    456.80    456.60    456.60    456.60    456.60    456.60    456.60
    456.60    456.60    456.60
```

continued overleaf

Table 2—*continued*

```
OPTIMAL INPUT XO3  FROM BRANCH 3  TO JUNCTION  5

   462.00    461.80    461.60    461.40    461.20    461.20    461.20    461.20
   461.20    461.20    461.20

OPTIMAL INPUT XO4  FROM BRANCH 4  TO JUNCTION  6

   472.00    471.80    471.60    471.40    471.20    471.00    470.80    470.60
   470.60    470.60    470.60

 "OPTIMAL SOLUTION OF THE SYSTEM"

 "STAGE  PIPE-DIAM  UP-ELEV  DECIS  DOWN-ELEV"

 "MAIN SERIAL PROCESS"
    9       .98      492.00   5.00   487.00
    8      1.13      487.00   8.00   479.00
    7      1.24      479.00   7.00   472.00
    6      1.77      472.00  10.00   462.00
    5      2.57      462.00   5.00   457.00
    4      2.79      457.00  10.00   447.00
    3      3.31      447.00   4.00   443.00
    2      3.68      443.00   3.00   440.00
    1      3.68      440.00   5.00   435.00

BRANCH  1  TO JUNCTION  3
    3      1.04      460.00   4.00   456.00
    2      1.15      456.00   4.00   452.00
    1      1.32      452.00   5.00   447.00

BRANCH  2  TO JUNCTION  4
    3      1.24      477.00  10.00   467.00
    2      1.64      467.00   5.00   462.00
    1      1.79      462.00   5.00   457.00

BRANCH  3  TO JUNCTION  5
    3      1.35      482.00   5.00   477.00
    2      1.36      477.00  10.00   467.00
    1      1.64      467.00   5.00   462.00

BRANCH  4  TO JUNCTION  6
    2      1.00      482.00   5.00   477.00
    1      1.32      477.00   5.00   472.00



TOTAL OPTIMAL RETURN IS   231418.

THE CPU TIME IN SECONDS IS      8.659
```

The computational complexity (both space and time complexity) of the two approaches, however, differed by more than 57%. The multi-converging branch algorithm required 19723 elementary operations with 11 discretizations of the state variables. See Esogbue and Warsi [14] for computational complexity analysis of converging branch systems. For comparative purposes, consider the DDDP approach which uses five discretizations at each stage in each iteration. The recursive equations would require three additions and one comparison at each junction and two additions and one comparison at all other stages. The total number of operations results in 34925, an astronomically higher number than our nonserial dynamic programming approach. The computer time requirement of the two approaches were also examined. The multi-converging branch approach required a total processing time (compilation time + execution time) of 20.5 CPU s (CYBER 855) while the discrete differential dynamic programming approach required 28.2 ~ 43.3 CPU s (IBM 360.75). The minimum cost solution indeed involved eleven iterations and a total processing time of 43.3 s for the DDDP approach while the inefficient DP approach took 113.7 s.

From the above results we conclude that the computational demands of the multi-converging branch algorithm is much less than the discrete differential dynamic programming approach, which is currently used in practice. Further, the computational superiority of our algorithm becomes more impressive when solving higher-dimensional (more branches and nodes per branch and main chain) and more complex (the structure of convergence as illustrated in Fig. 9) systems. Finally, global optimality is assured in all cases and the application is not restricted to special cost functions nor specially structured hydraulic systems as in the discrete deterministic dynamic programming model.

REFERENCES

1. R. E. Bellman and S. E. Dreyfus, *Applied Dynamic Programming*. Princeton Univ. Press, N.J. (1962).
2. A. O. Esogbue, Dynamic programming algorithms and analyses for nonserial networks, Technical Report, J-83-3, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Ga (1983).
3. R. E. Larson and W. G. Keckler, Applications of dynamic programming to water resources systems. *Proc. IFAC Haifa Conf. Computer Control of Natural Resources and Public Utilities*, Haifa, Israel (1967).
4. W. A. Hall and R. W. L. Shephard, Optimum operations for planning of a complex water resources system, Contribution No. 122, UCLA Water Resources Center, Los Angeles (1967).
5. L. Becker and W. G. Yeh, Optimization of real time operation of a multiple reservoir system. *Wat. Resour. Res.* **10,** 1107–1112 (1974).
6. L. Becker, W. G. Yeh, D. Fults and D. Sparks, Operations models for central valley project. *ASCE Jl Wat. Resour. Plann. Mgmt Div.* **102,** 101–115 (1976).
7. W. G. Yeh, L. Becker and W. S. Chu, Optimization of real-time hourly operations of a complex, multiple purpose reservoir system. UCLA Engineering Report No. UCLA-ENG. 7807, Los Angeles, Calif. (1978).
8. W. G. Yeh, L. Becker and W. S. Chu, Real time hourly reservoir operation. *ASCE Jl Wat. Resour. Plann. Mgmt Div.* **105,** 187–203 (1978).
9. Q. Martin, Minimum cost capacity expansion of a linear water conveyance pipeline. *ORSA/TIMS Bull.* No. 3, San Francisco, Calif. May (1977).
10. W. H. Tang, L. W. Mays and B. C. Yen, Optimal risk based design of stormed sewer networks. *45th Joint Nat. Meet. TIMS and ORSA*, Boston, Mass., 22–24 April (1974).
11. L. W. Mays and B. C. Yen, Optimal cost design of branched sewer systems. *Water Resour. Res.* **11**(1), 37–47 (1975).
12. V. T. Chow, D. R. Maidment and G. W. Tauxe, Computer time and memory requirements for DP and DDDP in water resources systems analysis. *Wat. Resour. Res.* **11**(5), 621–628 (1975).
13. R. Aris, G. L. Nemhauser and D. J. Wilde, Optimization of multistage cycle and branching systems by serial procedures. *A.ICh.E. Jl* **10**(6), 913–919 (1964).
14. A. O. Esogbue and N. A. Warsi, A high-level algorithm for diverging and converging branch nonserial dynamic programming systems. *Comput. Math. Applic.* **12A**(6), 719–732 (1986).